Title	Database Analyzer Manual 6.0
File Name	Database Analyzer Manual 6.doc
Category	Development & Analysis Tools
Version	6.0
Authors	Zoran Kukoljac
Contact	info@mallocinc.com
Created	December 21, 2016

Table of Contents

About Database Analyzer

1. The idea

The idea behind Database Analyzer is to use knowledge invested in database design to generate code that can be used in development process. Software intelligently predicts possible scenarios that might be needed in the future development.

Database Analyzer provides a set of tools that give a huge jump start in development process by:

- Generating comprehensive database analysis reports which can be used to quickly understand database structure and the data.
- Generating Java code that will be needed in development process

2. Solution

Database Analyzer (DBA) is a software package that analyzes database configuration, database structure, and data. DBA can then use that information to do the following:

- Generate reports where this information is structured in a specific way which helps to understand how the database is configured and structured. Most importantly, it helps to understand how data is stored and used in the database. It can help identify issues with the data model, corrupt data, and areas where improvements could be made in order to improve the overall system performance and to save resources. It is a valuable tool on ETL projects when:
    - 1) Data mapping or data dictionary has to be created and
    - 2) Database and its data has to be analyzed and understood. This tool could be used by developers, DBAs, Business Analysts, Data Analysts, System Architects, and anyone who is involved in development, analysis, or maintenance of the database.
- Generate Java code which is used to manipulate data and database structure. It provides a huge head start in the development process and makes maintenance easier and more efficient. For more information, see chapter “GDAO Development Methodology”.
- Generate other useful scripts for database and data management. For example, it can generate a script to drop all tables in the database based on their referential integrity (child tables have to be dropped before parent tables). Scripts like these are useful in development and testing environments where developers have limited privileges in the database. Some examples will be provided in this document.

The functionality and usage of the Database Analyzer

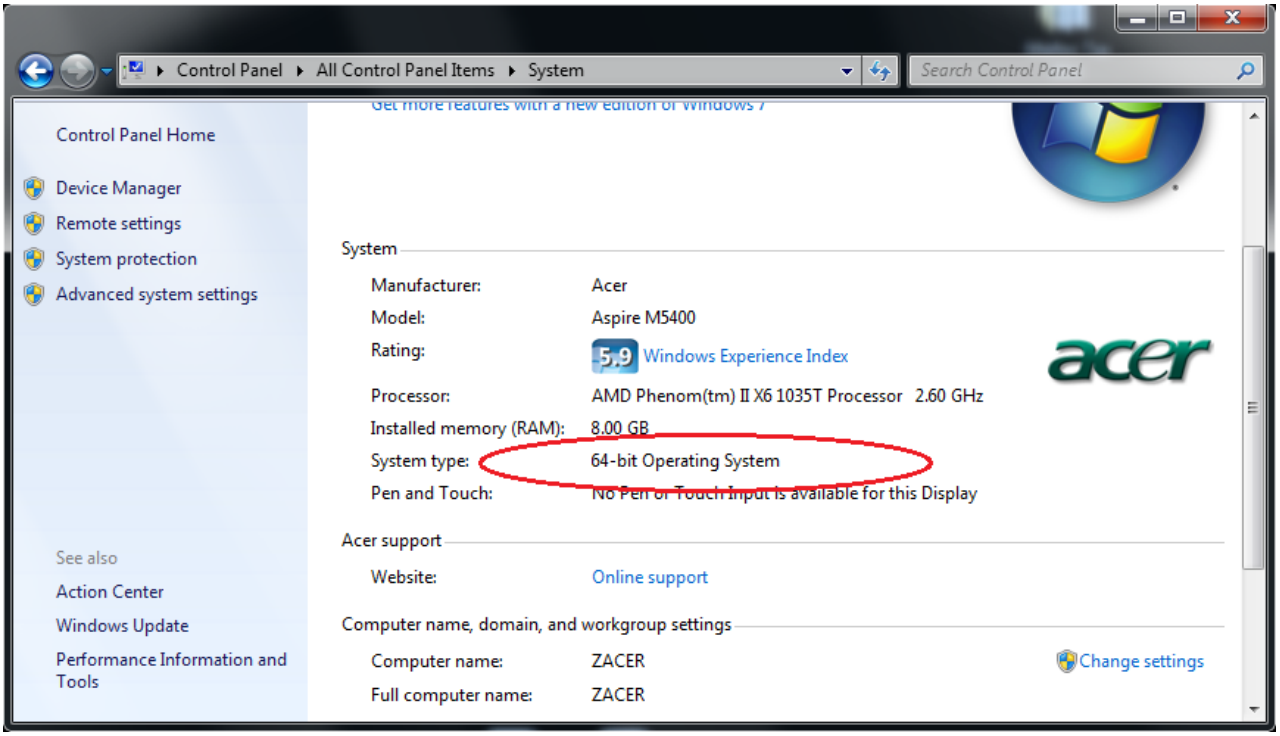
GDAO Development Methodology

"GDAO" stands for Generated Data Access Objects. Database Analyzer Software uses knowledge and logic stored in the database design to generate a number of Java classes for each database Table and View which helps developer to manipulate their structure and data. It vastly improves software development productivity when it comes to the development of all Java components that access databases. Moreover, the generated code is uniform and well-structured with plenty of comments so maintenance of the code is very efficient.

Installing Database Analyzer

3. Download Database Analyzer Software

First you need to verify the type of Windows Operating System (32-bit or 64-bit) you are using. You can find that information in "Control Panel" → "System".


Control Panel ▸ All Control Panel Items ▸ System
Search Control Panel
Control Panel Home
Device Manager
Remote settings
System protection
Advanced system settings
See also
Action Center
Windows Update
Performance Information and Tools
System
Manufacturer: Acer
Model: Aspire M5400
Rating: 5.9 Windows Experience Index
Processor: AMD Phenom(tm) II X6 1035T Processor 2.60 GHz
Installed memory (RAM): 8.00 GB
System type: 64-bit Operating System
Pen and Touch: No Pen or Touch Input is available for this Display
Acer support
Website: Online support
Computer name, domain, and workgroup settings
Computer name: ZACER
Full computer name: ZACER
Change settings

Download Database Analyzer software for your platform:

Version	Platform	Link
6.0	Windows 32 bit	http://downloads.mallocinc.com/dba60_32.zip
6.0	Windows 64 bit	http://downloads.mallocinc.com/dba60_64.zip
5.3	Windows 32 bit	http://www.mallocinc.com/DBAnalyzerR12010132.exe
5.3	Windows 64 bit	http://www.mallocinc.com/DBAnalyzerR12010164.exe

4. Install Software

Note that names of directories are arbitrary and are given here just as an example. You can choose another location where Database Analyzer software will be installed.

Step #1:

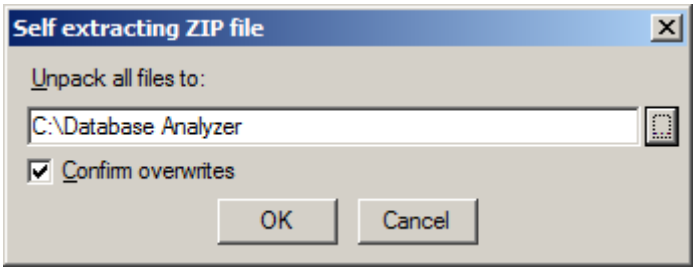
Create directory:

C:\Database Analyzer

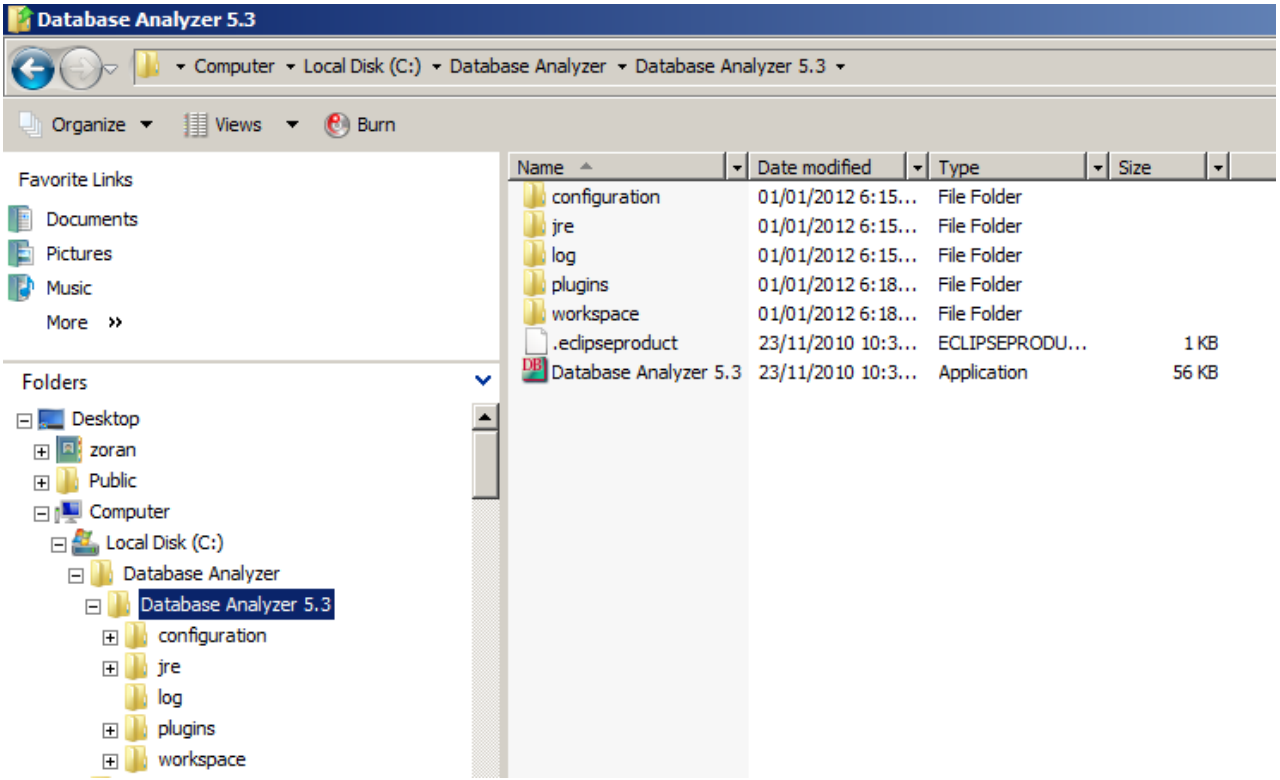
Step #2:

Unzip the DBA archive. If it is self extracting archive, do the following:

Run “InstallDatabaseAnalyzerR12010132.exe” and choose “C:\Database Analyzer” as a destination directory.



After extracting the downloaded file, your directory tree may look something like the picture shown below.



You are now ready to run Database Analyzer Application by executing:

c:\Database Analyzer\Database Analyzer 5.3\Database Analyzer 5.3.exe

Note:

Different versions of Database Analyzer software can co-exist and they can be installed in the same directory:

c:\Database Analyzer

For example:

c:\Database Analyzer\Database Analyzer 5.3

c:\Database Analyzer\Database Analyzer 6.0

Running Database Analyzer

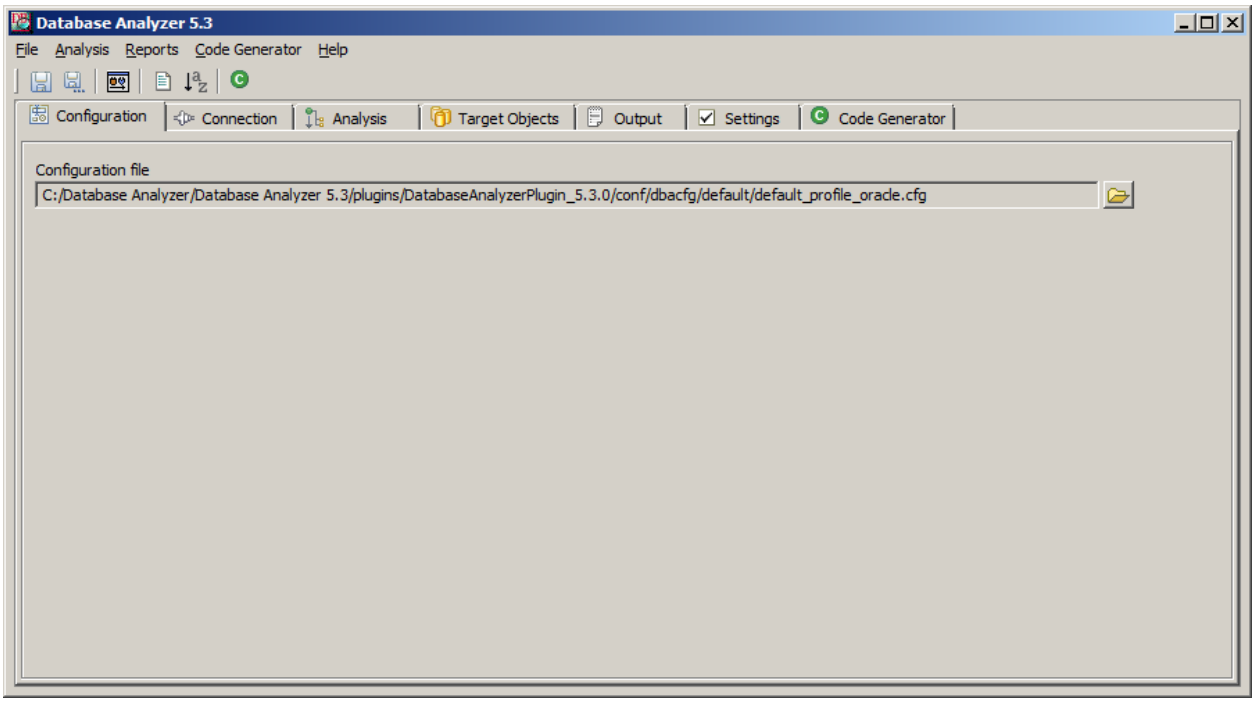
5. Configuration Tab

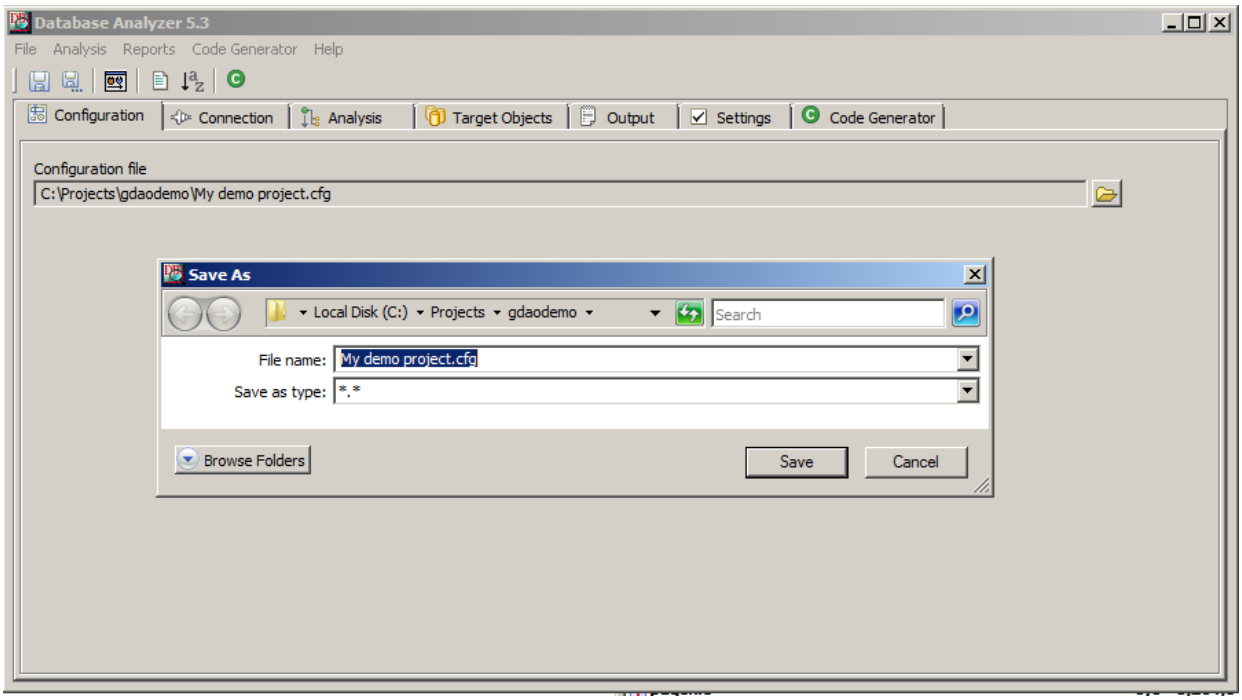
When you run Database Analyzer, you will get the screen where you choose the configuration file for your project. DBA Configuration file has information about certain projects:

1. Database connection
2. What Database Objects should be analyzed and what sample data should be extracted
3. Destination and packages for Java code that will be generated
4. Other information which will be demonstrated through examples

Configuration files are flat files that can be shared by many users.

The Default configuration file has parameters set for the GDAO Demo project. Before you start changing these parameters, you should save the configuration file under a different name and location. An example is shown on the screen below:



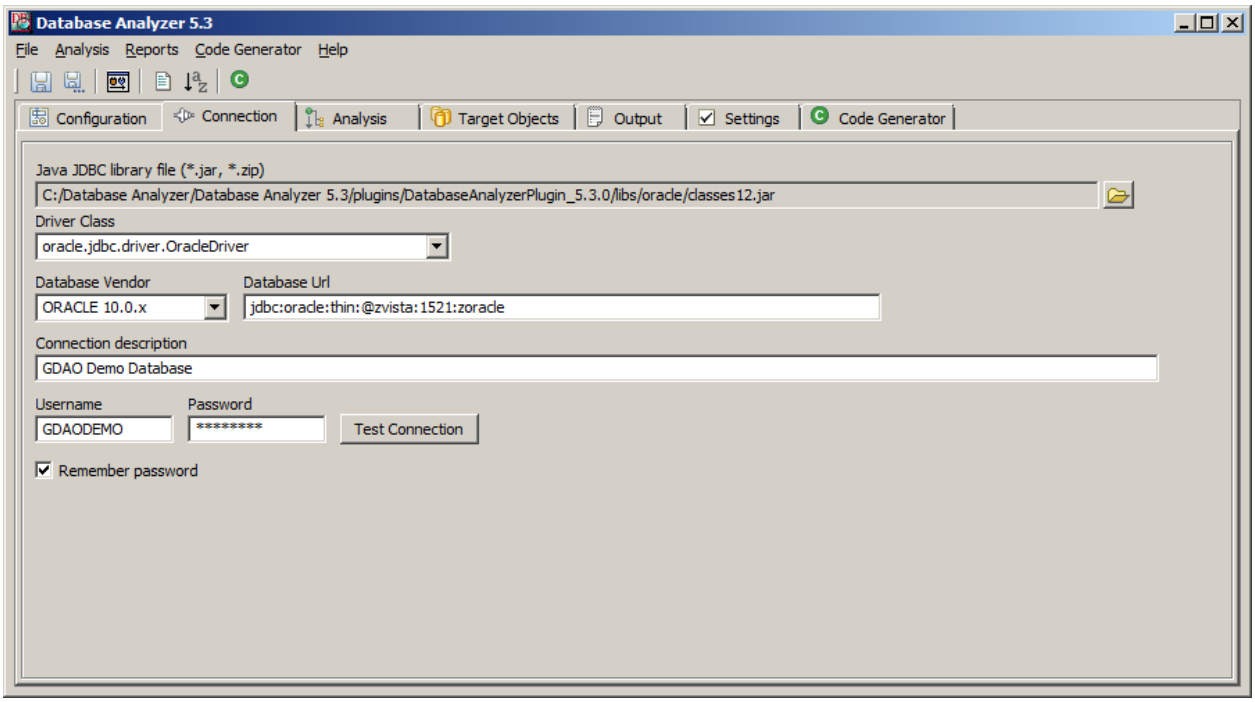

Database Analyzer 5.3
File Analysis Reports Code Generator Help
Configuration
Connection
Analysis
Target Objects
Output
Settings
Code Generator
Configuration file
C:\Projects\gdaodemo\My demo project.cfg
Save As
Local Disk (C:) Projects gdaodemo
Search
File name:
My demo project.cfg
Save as type:
.
Browse Folders
Save
Cancel

6. Connection Tab

On the “Connection” tab, you should change parameters to match your own server and database name, port, username, and password. You can also change the JDBC library and driver being used.

Use the “Test Connection” button to make sure you entered the correct parameters and you can connect to the database.

It is recommended that you download the most recent JDBC libraries from the vendor web site.



1. Supported Databases

Database Analyzer v6.0 can analyze and produce reports for the following databases:

1. Oracle
2. DB2
3. Sybase
4. Microsoft SQL Server
5. Access

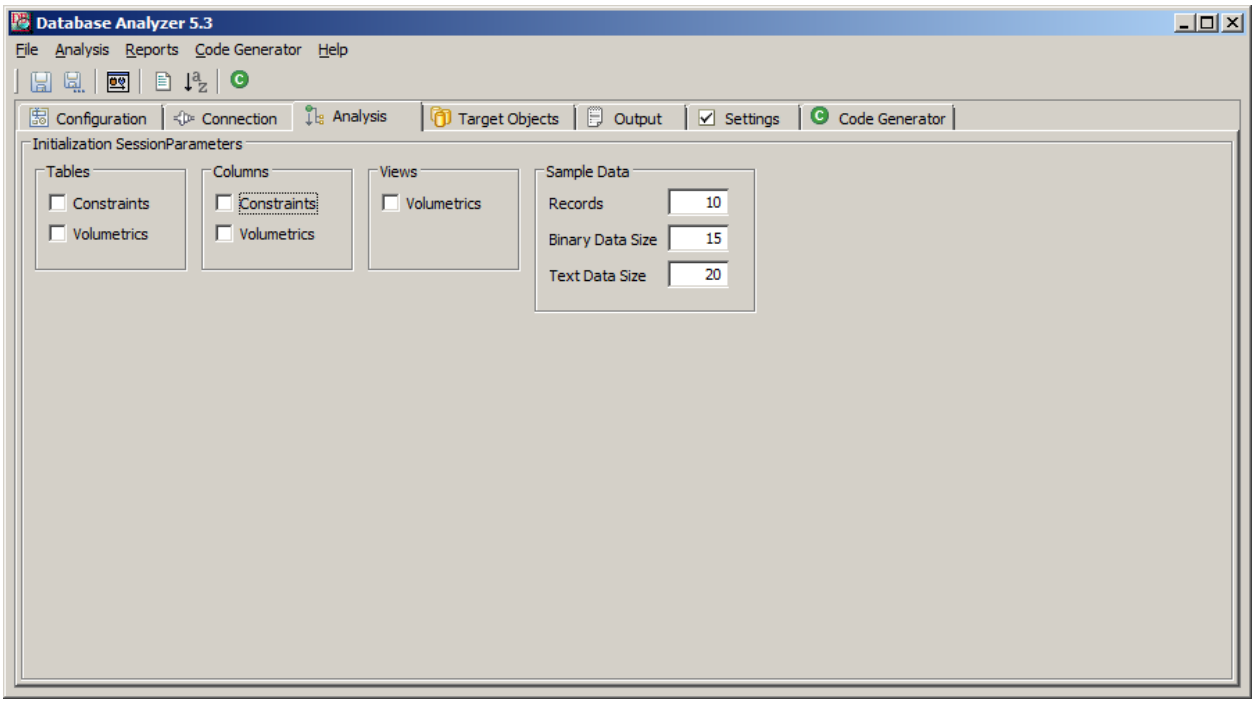
Code Generator feature of Database Analyzer v6.0supports the following databases:

1. Oracle

7. Analysis Tab

On the “Analysis” tab, specify what elements of the database should be analyzed. If you are not interested in data profiling, you should deselect all the “Constraints” and “Volumetric” check boxes since that operation can take a long time for large databases.

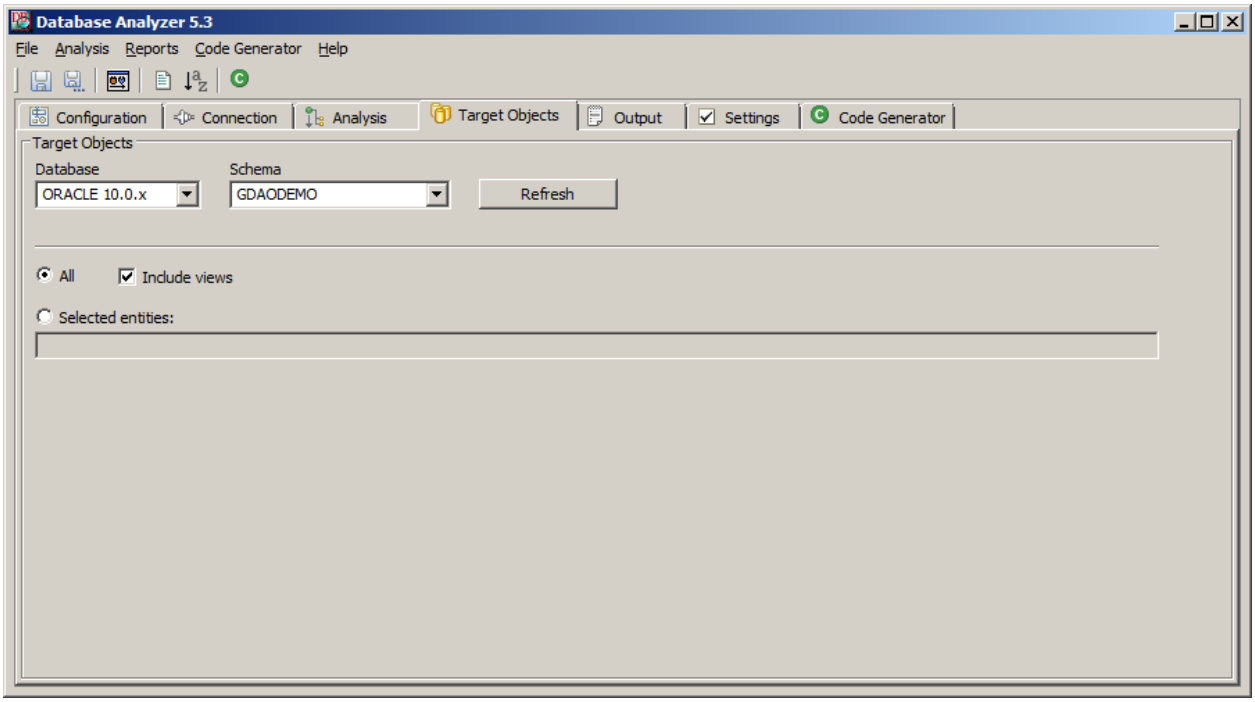
“Constraints” and “Volumetric” check boxes should be checked only when we are analyzing the database structure and when we are doing data profiling in order to generate reports. This will be explained under the “Database Analysis” chapter.



8. Target Objects Tab

On the “Target Objects” tab, select which database schema will be analyzed and which object within that schema should be analyzed.

You can connect as one user, but analyze objects that belong to another user providing that your first user has proper privileges. Also, you can specify a comma separated list of tables and views that you want to analyze. This is handy if only a few objects in a large database are needed to be analyzed.



1. Press the “Refresh” button to get the list of schemas in your database
2. For a database, choose “DEFAULT_DATABASE” and for a schema, choose the one that you would like to analyze. For GDAO Demo sample database, that would be the database user where new objects are created.

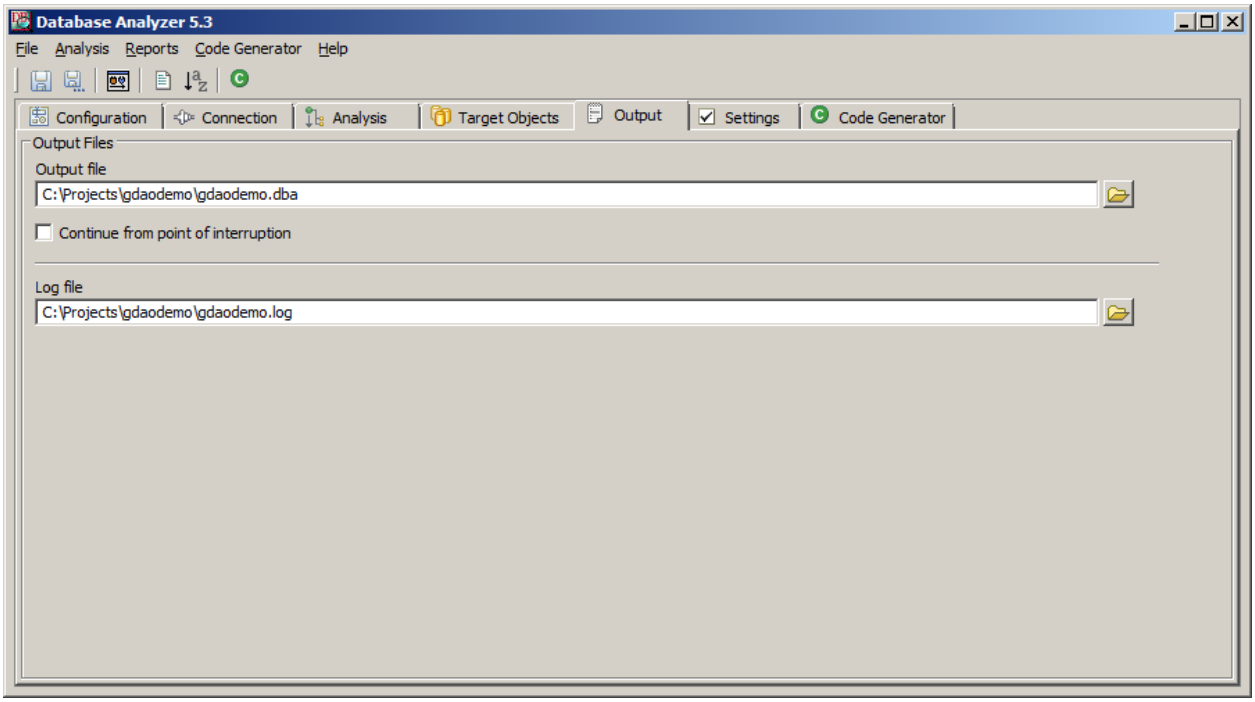
   Explanation for “DEFAULT_DATABASE:”
   For databases which support more than one database per Database Server, such as Sybase, you will have a drop down list with available databases. Since one Oracle Server Instance can support only one database, “DEFAULT_DATABASE” should be used.

9. Output Tab

On “Output” tab, you need to specify where Database Analyzer will store its files:

1. Analysis file
2. Log file

Note that directories where these files will be stored have to be created before the database analysis is run. In our example, this is the directory:
C:\Projects\gdaodemo

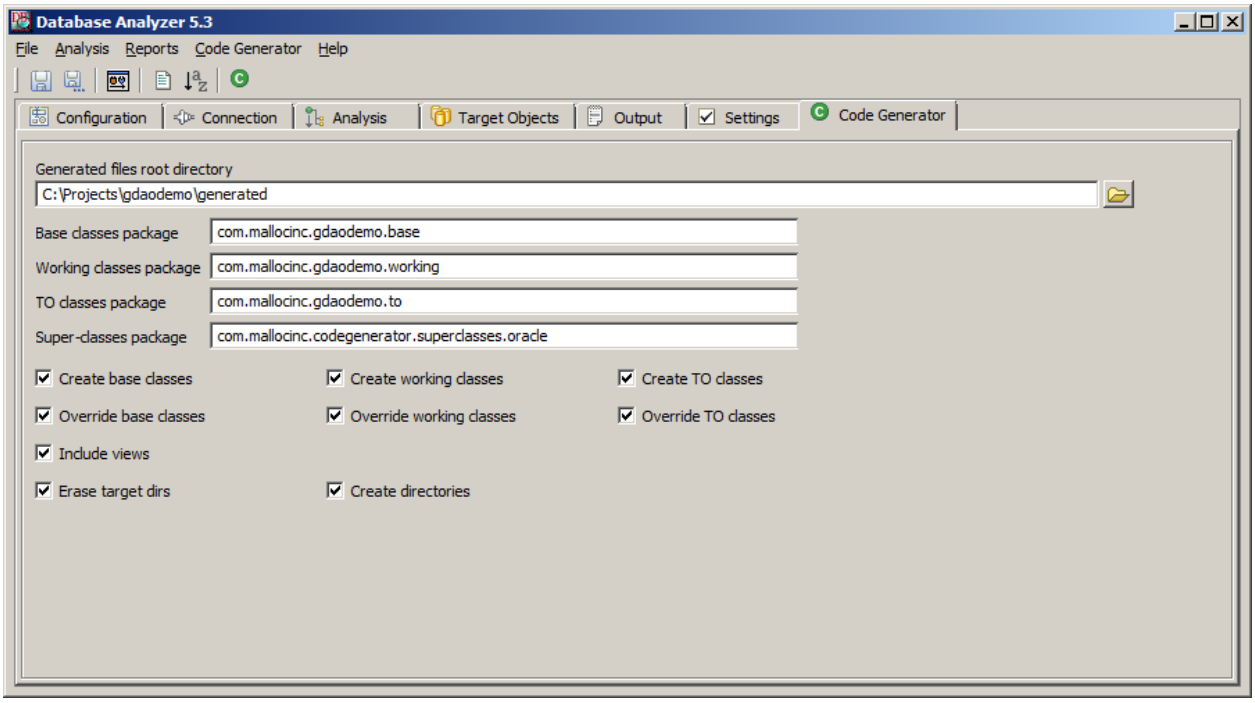


10. Code Generator Tab

The tab “Code Generator” contains all the information required to generate a java code based on the most recent database analysis.

Screen Field	Description
Generated files root directory	This is the directory under which all codes are generated. Note that the entire content of this directory is erased if the “Erase target dir” checkbox is selected. For this reason, it is strongly recommended that you don’t put any manually created or changed codes under this directory.
Base classes package	This is the package where GDAO Base classes are stored. Note that these classes should never be used directly in your programs. These classes are named in all “UPPERCASE” letters as a last line of defense against manual change.
Working classes package	These classes should be used to access the database.
TO classes package	These lightweight classes should be used to transfer data.
Super-classes package	This is the package with GDAO Super Classes distributed in both “gdao.jar” and “gdaosuper.jar”. These are the classes required for the GDAO Generated code to run. This location should never be changed unless you want to create your own super classes.

For more information about the relationship between the GDAO classes, check section “GDAO Architecture”.

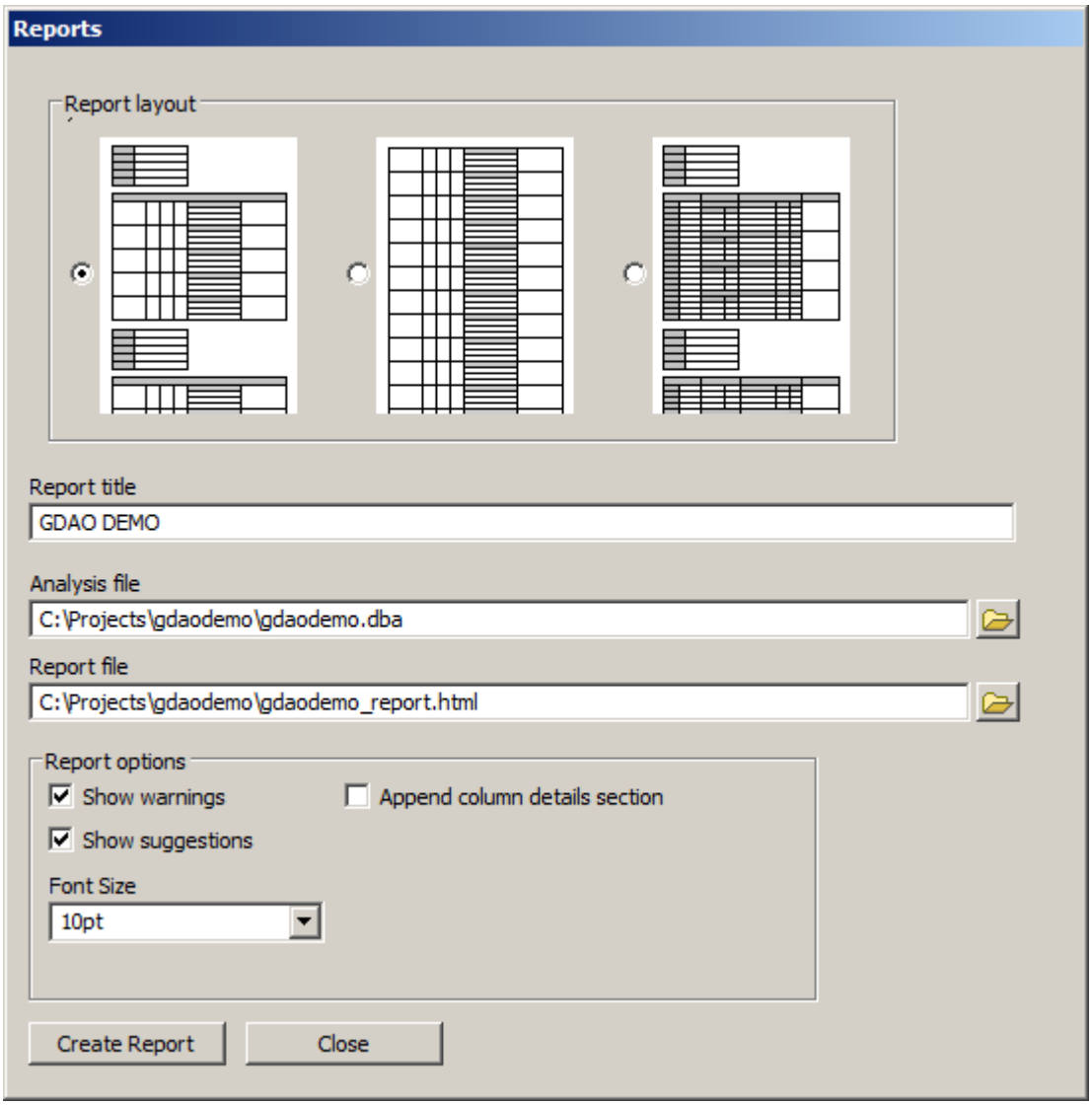


Locations of the “jar” files required by GDAO generated Java code:

- c:\Database Analyzer\Database Analyzer 5.3\plugins\DatabaseAnalyzerPlugin_5.3.0\gdao.jar
- c:\Database Analyzer\Database Analyzer 5.3\plugins\DatabaseAnalyzerPlugin_5.3.0\gdaosuper.jar

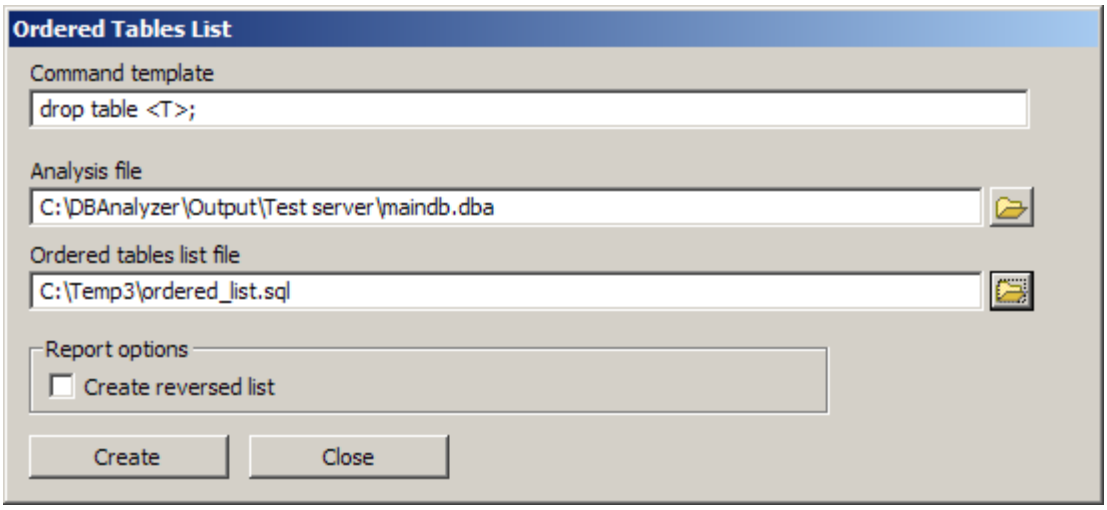
11. Creating Reports

On the “Create Reports” screen, you can create Data Profiling screens.
A sample of reports can be found on the Malloc Inc web site:



12. Creating ordered tables lists

This feature is used to produce an ordered table list depending on the relationship between the tables. For example, if you want to drop table by table, you would use this method to create a list of tables ordered in the way tables can be dropped, meaning that tables with no children will be dropped first.



This method creates another list with the reverse order from the list created by method "getOrderedTableList". For example, this option can be used to create tables.

File “C:\Temp3\ordered_list.sql” will be created with the following content (see the data model on the next page):

```
drop table TRANSACTION_ITEM;
drop table USER_PROFILE;
drop table CURRENCY_CODE;
drop table ROLE_PERMISSION;
drop table PERMISSION;
drop table USER_ROLE;
drop table ROLE;
drop table PAYMENT;
drop table PAYMENT_INSTRUMENTS;
drop table LOGIN_HISTORY;
drop table TRANSACTION;
drop table USERS;
drop table PERSON;
drop table ORGANIZATION;
drop table ADDRESS;
drop table ITEM;
```

Note that database views are ignored when the ordered tables list is created.

Starting New Development Project

Database Analyzer can be used in all phases of software development.

1. Analysis

1. Gather requirements
2. Feasibility study
3. Capacity planning
4. Create user cases
5. Create system architecture

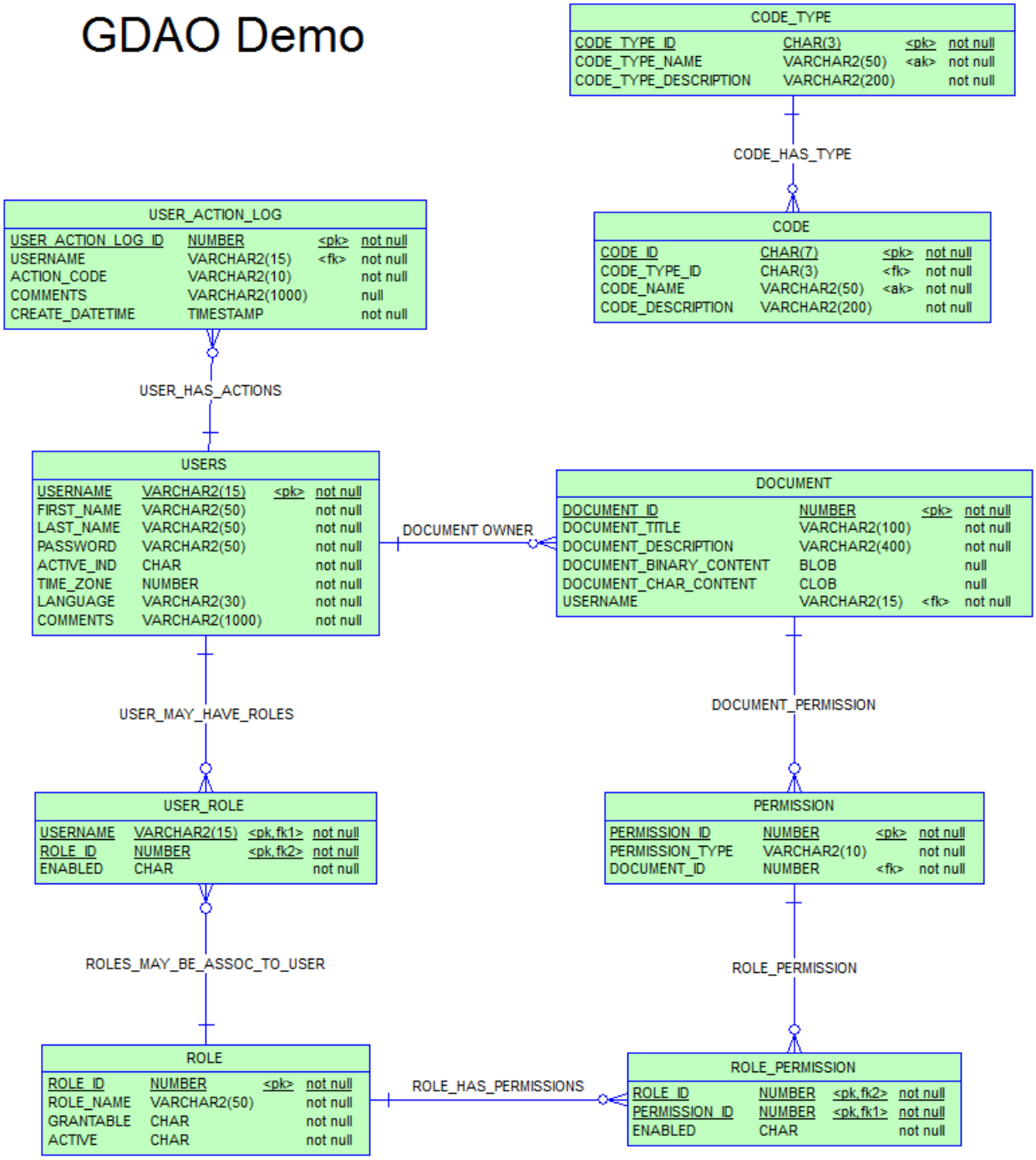
Activities where Database Analyzer can be used:

1. Proof of concept. This is the development of the modules that will be used to confirm that this concept will work in the new production environment.
2. Analysis of the Legacy Databases. It is needed in case if the database or data migration is required in a newly developed system.

2. Design

1. **Design the Database Model**

GDAO Demo


CODE_TYPE
CODE_TYPE_ID CHAR(3) <pk> not null
CODE_TYPE_NAME VARCHAR2(50) <ak> not null
CODE_TYPE_DESCRIPTION VARCHAR2(200) not null
CODE_HAS_TYPE
CODE
CODE_ID CHAR(7) <pk> not null
CODE_TYPE_ID CHAR(3) <fk> not null
CODE_NAME VARCHAR2(50) <ak> not null
CODE_DESCRIPTION VARCHAR2(200) not null
USER_ACTION_LOG
USER_ACTION_LOG_ID NUMBER <pk> not null
USERNAME VARCHAR2(15) <fk> not null
ACTION_CODE VARCHAR2(10) not null
COMMENTS VARCHAR2(1000) null
CREATE_DATETIME TIMESTAMP not null
USER_HAS_ACTIONS
USERS
USERNAME VARCHAR2(15) <pk> not null
FIRST_NAME VARCHAR2(50) not null
LAST_NAME VARCHAR2(50) not null
PASSWORD VARCHAR2(50) not null
ACTIVE_IND CHAR not null
TIME_ZONE NUMBER not null
LANGUAGE VARCHAR2(30) not null
COMMENTS VARCHAR2(1000) not null
DOCUMENT OWNER
DOCUMENT
DOCUMENT_ID NUMBER <pk> not null
DOCUMENT_TITLE VARCHAR2(100) not null
DOCUMENT_DESCRIPTION VARCHAR2(400) not null
DOCUMENT_BINARY_CONTENT BLOB null
DOCUMENT_CHAR_CONTENT CLOB null
USERNAME VARCHAR2(15) <fk> not null
USER_MAY_HAVE_ROLES
DOCUMENT_PERMISSION
USER_ROLE
USERNAME VARCHAR2(15) <pk,fk1> not null
ROLE_ID NUMBER <pk,fk2> not null
ENABLED CHAR not null
PERMISSION
PERMISSION_ID NUMBER <pk> not null
PERMISSION_TYPE VARCHAR2(10) not null
DOCUMENT_ID NUMBER <fk> not null
ROLES_MAY_BE_ASSOC_TO_USER
ROLE_PERMISSION
ROLE
ROLE_ID NUMBER <pk> not null
ROLE_NAME VARCHAR2(50) not null
GRANTABLE CHAR not null
ACTIVE CHAR not null
ROLE_HAS_PERMISSIONS
ROLE_PERMISSION
ROLE_ID NUMBER <pk,fk2> not null
PERMISSION_ID NUMBER <pk,fk1> not null
ENABLED CHAR not null

3. Development

1. Create Database Objects

```
/*==============================================================*/
/* Database name:  ZORACLE                                      */
/* DBMS name:      ORACLE Version 11g                           */
/* Created on:     30/12/2011 11:39:38 PM                       */
/*==============================================================*/


/*==============================================================*/
/* Table: CODE                                                  */
/*==============================================================*/
create table CODE  (
   CODE_ID              CHAR(7)                         not null,
   CODE_TYPE_ID         CHAR(3)                         not null,
   CODE_NAME            VARCHAR2(50)                    not null,
   CODE_DESCRIPTION     VARCHAR2(200)                   not null,
   constraint XPKCODE primary key (CODE_ID),
   constraint XAK1CODE_NAME unique (CODE_NAME)
);

/*==============================================================*/
/* Table: CODE_TYPE                                             */
/*==============================================================*/
create table CODE_TYPE  (
   CODE_TYPE_ID         CHAR(3)                         not null,
   CODE_TYPE_NAME       VARCHAR2(50)                    not null,
   CODE_TYPE_DESCRIPTION VARCHAR2(200)                   not null,
   constraint XPKCODE_TYPE primary key (CODE_TYPE_ID),
   constraint XAK1CODE_TYPE unique (CODE_TYPE_NAME)
);

/*==============================================================*/
/* Table: DOCUMENT                                              */
/*==============================================================*/
create table DOCUMENT  (
   DOCUMENT_ID          NUMBER                          not null
      constraint SYS_C009385 check ("DOCUMENT_ID" IS NOT NULL) not deferrable,
   DOCUMENT_TITLE       VARCHAR2(100)                   not null
      constraint SYS_C009387 check ("DOCUMENT_TITLE" IS NOT NULL) not deferrable,
   DOCUMENT_DESCRIPTION VARCHAR2(400)                   not null,
   DOCUMENT_BINARY_CONTENT BLOB,
   DOCUMENT_CHAR_CONTENT CLOB,
   USERNAME             VARCHAR2(15)                    not null,
   constraint PK_DOCUMENT primary key (DOCUMENT_ID)
);

/*==============================================================*/
/* Table: PERMISSION                                            */
/*==============================================================*/
create table PERMISSION  (
```

```
   PERMISSION_ID        NUMBER                          not null
      constraint SYS_C009408 check ("PERMISSION_ID" IS NOT NULL) not deferrable,
   PERMISSION_TYPE      VARCHAR2(10)                    not null,
   DOCUMENT_ID          NUMBER                          not null,
   constraint PK_PERMISSION primary key (PERMISSION_ID)
);

/*==============================================================*/
/* Table: ROLE                                                  */
/*==============================================================*/
create table ROLE  (
   ROLE_ID              NUMBER                          not null
      constraint SYS_C009414 check ("ROLE_ID" IS NOT NULL) not deferrable,
   ROLE_NAME            VARCHAR2(50)                    not null
      constraint SYS_C009415 check ("ROLE_NAME" IS NOT NULL) not deferrable,
   GRANTABLE            CHAR                            not null,
   ACTIVE               CHAR                            not null,
   constraint PK_ROLE primary key (ROLE_ID)
);

/*==============================================================*/
/* Table: ROLE_PERMISSION                                       */
/*==============================================================*/
create table ROLE_PERMISSION  (
   ROLE_ID              NUMBER                          not null
      constraint SYS_C009417 check ("ROLE_ID" IS NOT NULL) not deferrable,
   PERMISSION_ID        NUMBER                          not null
      constraint SYS_C009418 check ("PERMISSION_ID" IS NOT NULL) not deferrable,
   ENABLED              CHAR                            not null,
   constraint XPKROLE_PERMISSION primary key (ROLE_ID, PERMISSION_ID)
);

/*==============================================================*/
/* Table: USERS                                                 */
/*==============================================================*/
create table USERS  (
   USERNAME             VARCHAR2(15)                    not null
      constraint SYS_C009429 check ("USERNAME" IS NOT NULL) not deferrable,
   FIRST_NAME           VARCHAR2(50)                    not null,
   LAST_NAME            VARCHAR2(50)                    not null,
   PASSWORD             VARCHAR2(50)                    not null
      constraint SYS_C009430 check ("PASSWORD" IS NOT NULL) not deferrable,
   ACTIVE_IND           CHAR                            not null
      constraint SYS_C009431 check ("ACTIVE_IND" IS NOT NULL) not deferrable,
   TIME_ZONE            NUMBER                          not null,
   LANGUAGE             VARCHAR2(30)                    not null,
   COMMENTS             VARCHAR2(1000)                  not null,
   constraint PK_USERS primary key (USERNAME)
);

/*==============================================================*/
/* Table: USER_ACTION_LOG                                       */
/*==============================================================*/
create table USER_ACTION_LOG  (
```

```
   USER_ACTION_LOG_ID   NUMBER                          not null
      constraint SYS_C009371 check ("USER_ACTION_LOG_ID" IS NOT NULL) not deferrable,
   USERNAME             VARCHAR2(15)                    not null,
   ACTION_CODE          VARCHAR2(10)                    not null
      constraint SYS_C009373 check ("ACTION_CODE" IS NOT NULL) not deferrable,
   COMMENTS             VARCHAR2(1000),
   CREATE_DATETIME      TIMESTAMP                       not null
      constraint SYS_C009466 check ("CREATE_DATETIME" IS NOT NULL) not deferrable,
   constraint PK_ACTION_LOG primary key (USER_ACTION_LOG_ID)
);

/*==============================================================*/
/* Table: USER_ROLE                                             */
/*==============================================================*/
create table USER_ROLE  (
   USERNAME             VARCHAR2(15)                    not null,
   ROLE_ID              NUMBER                          not null
      constraint SYS_C009439 check ("ROLE_ID" IS NOT NULL) not deferrable,
   ENABLED              CHAR                            not null,
   constraint PK_USER_ROLE primary key (ROLE_ID, USERNAME)
);

alter table CODE
   add constraint CODE_HAS_TYPE foreign key (CODE_TYPE_ID)
      references CODE_TYPE (CODE_TYPE_ID)
      not deferrable;

alter table DOCUMENT
   add constraint "DOCUMENT OWNER" foreign key (USERNAME)
      references USERS (USERNAME)
      not deferrable;

alter table PERMISSION
   add constraint DOCUMENT_PERMISSION foreign key (DOCUMENT_ID)
      references DOCUMENT (DOCUMENT_ID)
      not deferrable;

alter table ROLE_PERMISSION
   add constraint ROLE_HAS_PERMISSIONS foreign key (ROLE_ID)
      references ROLE (ROLE_ID)
      not deferrable;

alter table ROLE_PERMISSION
   add constraint ROLE_PERMISSION foreign key (PERMISSION_ID)
      references PERMISSION (PERMISSION_ID)
      not deferrable;

alter table USER_ACTION_LOG
   add constraint USER_HAS_ACTIONS foreign key (USERNAME)
      references USERS (USERNAME)
      not deferrable;

alter table USER_ROLE
   add constraint ROLES_MAY_BE_ASSOC_TO_USER foreign key (ROLE_ID)
```

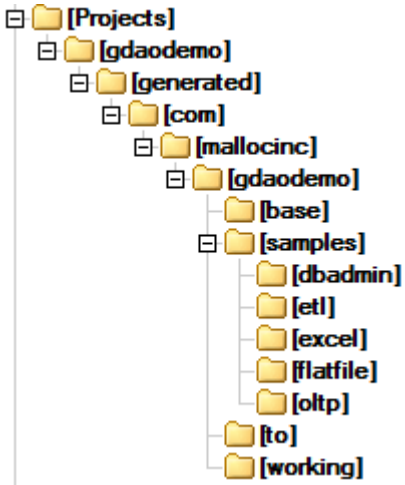
```
      references ROLE (ROLE_ID)
      not deferrable;

alter table USER_ROLE
   add constraint USER_MAY_HAVE_ROLES foreign key (USERNAME)
      references USERS (USERNAME)
      not deferrable;
```

2. Database Analyzer in the Development Process

1. Using Database Analyzer:
   a. Analyze Database
   b. Generate Code

When you generate code in the directory, "C:\Projects\gdaodemo\generated," your new directory structure will look something like the picture shown below:



All directories and files under "C:\Projects\gdaodemo\generated" are generated by the Database Analyzer. Note that the names of these directories are set up in the screen "Code Generator Tab" and they can be changed to fit your needs. Values in the example are used just to illustrate this concept.

Explanation of directory structure:

Directory	Description
com/mallocinc/gdaodemo/base/	Location where GDAO Base Classes are generated.
com/mallocinc/gdaodemo/to/	Location where GDAO TO (Transfer Object) Classes are generated.
com/mallocinc/gdaodemo/working/	Location where GDAO Working Classes are generated/
com/mallocinc/gdaodemo/samples/oltp	Sample codes that demonstrate how GDAO Generated code can be used for Online Transaction Processing.

	This would probably include examples that would be used most frequently in a typical software development project.
com/mallocinc/gdaodemo/samples/dbadmin	Sample codes that demonstrate how GDAO Generated code can be used for Database Administrative tasks such as creating and changing database objects.
com/mallocinc/gdaodemo/samples/etl	Sample codes that demonstrate how GDAO Generated code can be used for ETL (Data Extraction, Transformation, and Load) Processing.
com/mallocinc/gdaodemo/samples/excel	Sample codes that demonstrate how GDAO Generated code can be used to import and export data from and to Excel files.
com/mallocinc/gdaodemo/samples/flatfile	Sample codes that demonstrate how GDAO Generated code can be used to import and export data from and to flat files.

Sample Classes could be run “as is.” However, the recommended process is to use them as a catalog of codes to extract the parts you need and put them in your own classes. For example, if you need a code that selects data from the table, you should find it in Sample class, extract it, and put it in your own class.

In each of the Generated “Sample” classes, a “main()” method has a functionality to demonstrate how GDAO can be used (how the particular functionality should be invoked). A “main()” method demonstrates some other functionalities such as a connection to the database, getting parameters from Properties files, etc. You can always use your own methods to connect to the database and configure the application.

The development process using GDAO is demonstrated in various sample scenarios below.

Now you should include the following “jar” files to your project.

1. \Commons\commons-lang3-3.1.jar
2. \Commons\commons-io-2.1.jar
3. \Commons\commons-configuration-1.7.jar
4. \Commons\commons-collections-3.2.1.jar
5. \Commons\commons-logging-1.1.1.jar
6. \Excel\jxl.jar
7. \log4j\log4j-1.2.14.jar
8. \Oracle\Oracle11g\ojdbc6.jar
9. \Sybase\jconn3.jar
10. \Xerces_2_6_2\resolver.jar
11. \Xerces_2_6_2\xercesImpl.jar
12. \Xerces_2_6_2\xml-apis.jar
13. \db2\db2jcc.jar
14. \db2\db2jcc_license_cu.jar
15. \plugins\DatabaseAnalyzerPlugin_5.3.0\gdao.jar

All “jar” files except “gdao.jar” are available in the directory below; however, you can always try newer libraries from respective vendor sites:

```
c:\Database Analyzer\Database Analyzer 5.3\plugins\DatabaseAnalyzerPlugin_5.3.0\libs
```

3. **Template Constants**

Constant values in templates are set in this property file:

`\plugins\DatabaseAnalyzerPlugin_5.3.0\bin\templates\gdao_oracle.properties`

4. **Log File**

Location of Database Analyzer Application Log File:

`\log\dbadministrator.log`

Development Scenario for Using GDAO Generated Code

1. Creating Test Data

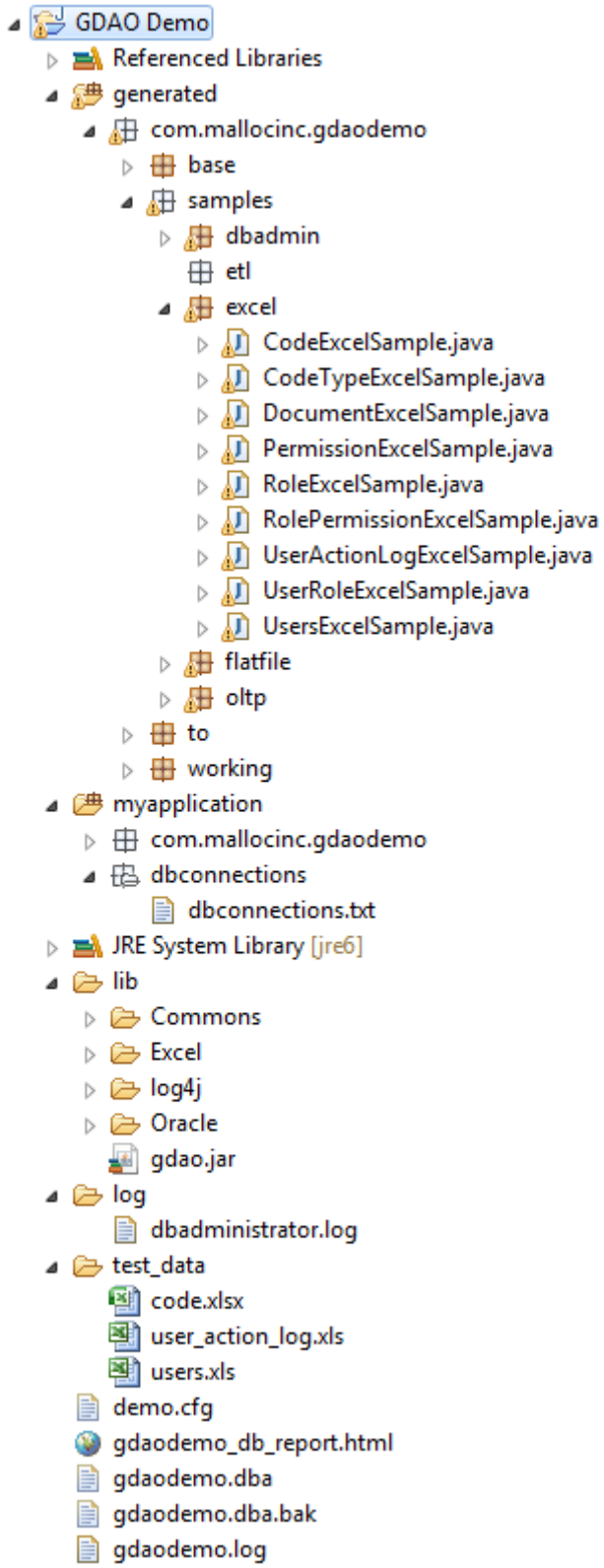
This example is going to demonstrate test data for tables USERS and USERS_ACTION_LOG. Data will be created in Excel Spreadsheets and then GDAO generated code will be used to load it into the database.

The easiest way is to create one spreadsheet per table:

1. The Spreadsheet will have a name with this naming convention *table_name* + ".xls".
2. The first row in the Spreadsheet will be a header row and it will be ignored during the load process.
3. The number of columns, their order, and data type should match columns in the respective database table.

Note that these rules are given here as examples. In your own project, you have the flexibility to change these rules and in that case, the generated code should be altered accordingly. Also, all destination directories that you are going to create in the process described below are arbitrary and you can choose your own locations.

When you are done with this setup process, your directory structure will look something like this:



In order to use the test file provided with this demo as well as convenient methods for connecting to the database, you need to do the following:

2. Create Your Application Code Directory

Create a directory "**myapplication**" where you are going to store your manually created code.

3. Edit "dbconnections.txt" File

Under your "**myapplication**" directory, create directory "**dbconnections**" and copy the provided "dbconnections.txt" to this new location. Edit "dbconnections.txt" to match your database parameters.

Using the main method of every "Sample" class, you will be able to connect to various databases by only selecting the database connection id that is defined in this file (see Sample class for details).

Remember that you can always code your own way to create a database connection. This functionality is provided for your convenience.

4. Copy test files to your directory

Copy test data to directory "c:\Projects\gdaodemo\test_data\".

You can find the Excel test file in directory:
c:\Database Analyzer\Database Analyzer 5.3\plugins\DatabaseAnalyzerPlugin_5.3.0

1. \bin\oracle\Code Generator Demo\test_data\excel_files\user_action_log.xls
2. \bin\oracle\Code Generator Demo\test_data\excel_files\users.xls

5. Edit Application Configuration File

Edit "c:\Projects\gdaodemo\demo.cfg" file.

```
#GDAO demo configuration file
db_connection_id=GDAODEMO
excel_test_files_directory=./test_data/
excel_file_suffix=xls
```

6. Run Sample Code

You are now ready to run your sample code. For example, if you run "UsersExcelSample", records will be loaded from excel file "users.xls" into the database table "USERS". Note that the "main()" method of sample class has other functions that are commented out. For example – extracting data from the database into the Excel file and updating database based on data in the Excel file. If you need to keep any of these functionalities, you can de-comment them in Sample class or copy the related code into your own class.

When you run this sample code, it should produce an output in the log file similar to this one:

```
2012-01-02 19:00:59,136 INFO  [main] UsersExcelSample -> Current Directory -> C:\Projects\gdaodemo
2012-01-02 19:00:59,214 INFO  [main] UsersExcelSample -> dbConnectionId = GDAODEMO
2012-01-02 19:00:59,214 INFO  [main] UsersExcelSample -> excelTestFilesDirectory = ./test_data/
2012-01-02 19:00:59,214 INFO  [main] UsersExcelSample -> excelFileSuffix = xls
2012-01-02 19:00:59,214 INFO  [main] GDAODatabaseConnectionFactory -> dbConnectionsFileURL:
[file:/C:/Projects/gdaodemo/bin/dbconnections/dbconnections.txt]
2012-01-02 19:00:59,214 INFO  [main] GDAODatabaseConnectionFactory -> dbConnectionsFileURL.getFile():
[/C:/Projects/gdaodemo/bin/dbconnections/dbconnections.txt]
2012-01-02 19:00:59,230 INFO  [main] FlatFileLoaderBase -> Current directory: 'C:\Projects\gdaodemo'
2012-01-02 19:00:59,230 INFO  [main] FlatFileLoaderBase -> Input file       :
'C:\Projects\gdaodemo\bin\dbconnections\dbconnections.txt'
2012-01-02 19:00:59,230 INFO  [main] FlatFileLoaderBase -> ***************** Input file information
*******************
2012-01-02 19:00:59,230 INFO  [main] FlatFileLoaderBase -> Name:
C:\Projects\gdaodemo\bin\dbconnections\dbconnections.txt
2012-01-02 19:00:59,230 INFO  [main] FlatFileLoaderBase -> Size:    992 bytes
2012-01-02 19:00:59,230 INFO  [main] FlatFileLoaderBase -> Created: 2012-01-01 10:33:14.036
2012-01-02 19:00:59,230 INFO  [main] FlatFileLoaderBase -> Current directory: 'C:\Projects\gdaodemo'
2012-01-02 19:00:59,230 INFO  [main] FlatFileLoaderBase ->
************************************************************
2012-01-02 19:00:59,230 INFO  [main] DBConnectionsLoader -> Processing DB Connection file:
[/C:/Projects/gdaodemo/bin/dbconnections/dbconnections.txt]
2012-01-02 19:00:59,230 INFO  [main] DBConnectionsLoader -> ***** Found a match!: GDAODEMO ******
0. [    GDAODEMO]
1. [ GDAODEMO Database - Oracle 11g]
2. [    jdbc:oracle:thin:@zvista:1521:zoracle]
3. [         GDAODEMO]
4. [   GDAODEMO]
5. [     oracle.jdbc.driver.OracleDriver]
2012-01-02 19:00:59,230 INFO  [main] FlatFileLoaderBase -> File
'C:\Projects\gdaodemo\bin\dbconnections\dbconnections.txt' is processed in: 0 ms
Connection name: GDAODEMO
Connection description: GDAODEMO Database - Oracle 11g
Connection URL: jdbc:oracle:thin:@zvista:1521:zoracle
Username: GDAODEMO
Password: GDAODEMO
Driver: oracle.jdbc.driver.OracleDriver

2012-01-02 19:00:59,573 INFO  [main] GDAODatabaseConnectionFactory -> Connected to "GDAODEMO Database - Oracle
11g"
2012-01-02 19:00:59,573 INFO  [main] UsersExcelSample -> Test File = ./test_data/USERS.xls
2012-01-02 19:00:59,822 INFO  [main] UsersExcelSample -> Spreadsheet has 70 rows and 8 columns
zorank;Bruce;Kelly;daryl777;Y;5;ENGLISH;Test user 1
peter;Bryan;Macbeth;james777;N;3;FRENCH;Test user 2
john;Calvin;Macdonald;_roberto_;N;2;GERMAN;Test user 3
johana;Caroline;Bagley;laxuss;Y;-1;ENGLISH;Test user 4
jannah;Carolyn;Bagwell;pion;Y;3;FRENCH;Test user 5
daryl;Casey;Bailey;li0n;N;-10;GERMAN;Test user 6
bianka;Cathy;Bainbridge;cr0ok;N;12;ENGLISH;Test user 7
andrej;Charles;Baker;h00k;Y;11;FRENCH;Test user 8
zkukoljac;Christie;Ball;captainNemo;Y;0;GERMAN;Test user 9
myuser;Christy;Banahan;Groot10;N;4;ENGLISH;Test user 10
worklduser;Chuck;Bannon;diogen;N;4;FRENCH;Test user 11
sara;Cindy;Barnett;nancy22;Y;5;GERMAN;Test user 12
mara;Clark;Barrett;coyote6;Y;5;ENGLISH;Test user 13
bara;Claude;Barry;coyote10;N;5;FRENCH;Test user 14
shinami;Cliff;Barton;^ROWER;N;5;GERMAN;Test user 15
johan;Clint;Baskin;paddler^^;Y;6;ENGLISH;Test user 16
dan;Colin;Bates;Dijon%1;Y;6;FRENCH;Test user 17
jeffrey;Cory;Beattie;Ajilon;N;6;GERMAN;Test user 18
lui;Courtney;Beckett;pujilon?^;N;6;ENGLISH;Test user 19
zoran11;Craig;Beegan;56zeus;Y;7;POLISH;Test user 20
zoran12;Crystal;Begg;2bobo;N;8;DUCH;Test user 21
googleXz;Curt;Behan;1fred;N;8;DUCH;Test user 22
superman;Cynthia;Bell;fredex;Y;8;ENGLISH;Test user 23
supergirl;Dale;Bellew;biovax;Y;9;SPANISH;Test user 24
tiger;Dale;Bennett;trixus;N;8;SPANISH;Test user 25
peters;Dan;Bentley;triluim55;N;9;ENGLISH;Test user 26
fabio;Dana;Berry;vally44;Y;9;FRENCH;Test user 27
krakinho;Dana;Biggs;nelly;Y;9;GERMAN;Test user 28
antilope;Danielle;Black;xnelly;N;10;ENGLISH;Test user 29
mire;Danny;Blake;cycler;N;10;FRENCH;Test user 30
daryl777;Darlene;Blaney;askme1;Y;10;GERMAN;Test user 31
james777;Darren;Boden;123456;Y;10;ENGLISH;Test user 32
roberto;Darren;Boland;890865;N;11;FRENCH;Test user 33
laxus;Dave;Bolger;x23;N;5;GERMAN;Test user 34
pion;Dawn;Bolton;kajun;Y;3;ENGLISH;Test user 35
lion;Deanna;Bourke;kajun33;Y;2;FRENCH;Test user 36
crook;Debbie;Boyce;li0n;N;-1;GERMAN;Test user 37
hook;Debby;Boylan;cr0ok;N;3;ENGLISH;Test user 38
```

```
captain;Denise;Bowen;h00k;Y;-10;FRENCH;Test user 39
superuser;Dennis;Boyle;captainNemo;N;12;GERMAN;Test user 40
admin;Derek;Bradley;Groot11;N;11;ENGLISH;Test user 41
Administrator;Diana;Bradshaw;diogen;Y;0;FRENCH;Test user 42
root;Diane;Brady;nancy23;Y;4;GERMAN;Test user 43
diogen;Dillon;Brahney;coyote14;N;4;ENGLISH;Test user 44
nancy22;Dirk;Breheny;coyote18;N;5;POLISH;Test user 45
coyote6;Dominic;Brannigan;^ROWER;Y;5;DUCH;Test user 46
coyote10;Don;O'brannigan;paddler^^;Y;5;DUCH;Test user 47
ROWER;Donald;Breeden;Dijon%2;N;5;ENGLISH;Test user 48
paddler;Donna;Breen;Ajilon;N;6;SPANISH;Test user 49
Dijon;Doris;Brennan;pujilon?^;Y;6;SPANISH;Test user 50
Ajilon;Dorothy;Breslin;56zeus;Y;6;ENGLISH;Test user 51
pujilon;Doug;Bresnahan;2bobo;N;6;FRENCH;Test user 52
zeus;Douglas;Brewer;1fred;N;7;GERMAN;Test user 53
bobo;Dustin;Mcbride;fredex;Y;8;ENGLISH;Test user 54
fred;Dylan;O'brien;biovax;Y;8;FRENCH;Test user 55
fredex;Earl;Briody;trixus;N;8;GERMAN;Test user 56
biovax;Eddie;Broderick;triluim56;N;9;ENGLISH;Test user 57
trixus;Edgar;Brodigan;vally45;Y;5;FRENCH;Test user 58
triluim;Edwin;Brogan;nelly;N;3;GERMAN;Test user 59
vally;Eli;Brophy;xnelly;N;2;ENGLISH;Test user 60
nelly;Elizabeth;Brooks;cycler;Y;-1;FRENCH;Test user 61
xnelly;Ellen;Brown;askme2;Y;3;GERMAN;Test user 62
cycler;Elliot;Browne;1658274;N;-10;ENGLISH;Test user 63
askme1;Emily;O'bryant;2425683;N;12;FRENCH;Test user 64
A123456;Eric;Bryson;x24;Y;11;GERMAN;Test user 65
b890865;Erica;Buckley;kajun;Y;0;ENGLISH;Test user 66
x23;Erin;Burke;kajun34;N;4;FRENCH;Test user 67
kajun;Erin;Burnett;li0n;N;4;GERMAN;Test user 68
kajun33;Ernie;Burns;cr0ok;Y;5;ENGLISH;Test user 69
2012-01-02 19:01:00,182 INFO  [main] BaseDbEntityClass -> Total number of rows in TABLE USERS : 69
2012-01-02 19:01:00,182 INFO  [main] UsersExcelSample -> Inserted Records in TABLE USERS DB: 69. Errors: 0
2012-01-02 19:01:00,182 INFO  [main] UsersExcelSample -> Total row count in TABLE "USERS") is 69
```

Creating Database Reports

All reports are produced in HTML format and can be opened in all popular browsers. Reports are also editable – they can be imported into, for example, MS Word where the file can be converted in a different format, including Microsoft ".doc" format. Once the HTML file is imported into Word, user can add footers, headers, and page number as well as select paper orientation and basically use any MS Word feature to make the document look like any other company document. Some examples are provided below.

Special features:

- reports can be opened in any browser, no need for specialized software to read reports
- reports are editable
- provide information about the database structure and data
- provide warnings and suggestions about possible improvements of the data model
- easy to produce
- available in three different formats with options to include or exclude sections of the report

Sample MS Word Documents.

Classic report provides information about the database server, databases (Sybase and MS SQL Server support multiple databases under the same database server), schemas, tables and views, columns, and data.

Continuous report provides the same information as Classic report; however, in a format which is suitable for adding new columns to the document. This is particularly handy when users want to put some additional information such as data transformation rules in case of a mapping document.

Relationship report provides information about referential integrity in the database. This report allows the user to see which Primary Key columns are referenced by which foreign key columns and vice versa. Some popular data modeling tools cannot give information about parent columns, so this feature is a great addition to Case tools.

Common sections in all reports:

- Basic database server features
- Database server limitations
- Database server supporting features
- List of databases
- Detailed tables of information
- List of schemas in the database selected to be analyzed
- List of tables in databases and schema selected to be analyzed
- Detailed columns of the information section (at the end of the document)

1. Report sections

1. Basic database server features

Feature	Value
Connection Description	DBAnalyzer 1 Database on Tower
Product Name	Oracle
Database product name:	Oracle
Catalog term:	
Schema term:	schema
Identifier Quote String:	Oracle
Search String Escape:	//
"Extra" characters that can be used in unquoted identifier names (those beyond a-z, A-Z, 0-9 and _):	$#
Database Major Version:	Information not available
Database Minor Version:	
Database Product Version:	Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production With the Partitioning, OLAP and Data Mining options
Driver Major Version:	9
Driver Name:	Oracle JDBC driver
Driver Version:	9.2.0.1.0
Numeric Functions:	ABS, ACOS, ASIN, ATAN, ATAN2, CEILING, COS, EXP, FLOOR, LOG, LOG10, MOD, PI, POWER, ROUND, SIGN, SIN, SQRT, TAN, TRUNCATE
String Functions	ASCII, CHAR, CONCAT, LCASE, LENGTH, LTRIM, REPLACE, RTRIM, SOUNDEX, SUBSTRING, UCASE
Database URL	jdbc:oracle:thin:@192.168.1.80:1521:zoracle
User name used for this connection	DBANALYZER_1
Database is read only	No
Database is case sensitive	N
Data definition statement within a transaction forces the transaction to commit	Y
Database ignores a data definition statement within a transaction	N
String used to quote SQL identifiers	"

2. Database Server limitations

Feature	Value (zero means that there is no limit or the limit is not known)
Max Row Size:	2000
Max Row Size includes BLOB(s) (includes the SQL data types LONGVARCHAR and LONGVARBINARY)	Y
Max Table Name Length:	30
Max User Name Length:	30
Max Tables In Select	0
Maximum length (in hex characters) for a binary literal	1000
Maximum number of characters allowed in a catalog name	0
Maximum number of characters allowed for a character literal	2000
Maximum number of characters this database allows for a column name	30
Maximum number of columns this database allows in a GROUP BY clause	0
Maximum number of columns this database allows in an index	32
Maximum number of columns this database allows in an ORDER BY clause	0
Maximum number of columns this database allows in a SELECT list	0
Maximum number of columns this database allows in a table	1000
Maximum number of concurrent connections to this database that are possible	0
Maximum number of characters that this database allows in a cursor name	0
Maximum number of characters that this database allows in a cursor name	0
Maximum number of characters that this database allows in a procedure name	30
Maximum number of characters that this database allows in a schema name	30
Maximum number of characters this database allows in an SQL statement	65535
Maximum number of active statements to this database that can be open at the same time	0

List of schemas in the database selected to be analyzed

Schema DBANALYZER_1	Table	Type	Rows	Columns
1.	ADDRESS	TABLE	300	10
2.	CURRENCY_CODE	TABLE	43	3
3.	ITEM	TABLE	2,041	7
4.	LOGIN_HISTORY	TABLE	1,987	4
5.	ORGANIZATION	TABLE	42	5
6.	PAYMENT	TABLE	5,187	4
7.	PAYMENT_INSTRUMENTS	TABLE	31	3
8.	PERMISSION	TABLE	8	5
9.	PERSON	TABLE	108	8
10.	ROLE	TABLE	10	4
11.	ROLE_PERMISSION	TABLE	80	5
12.	TRANSACTION	TABLE	4,318	4
13.	TRANSACTION_ITEM	TABLE	989	3
14.	USERS	TABLE	191	5
15.	USER_PROFILE	TABLE	19	3
16.	USER_ROLE	TABLE	10	3

Database server supporting features

Feature	Supported
Mixed Case Identifiers	N
Mixed Case Quoted Identifiers	Y
Alter Table With Add Column	Y
Alter Table With Drop Column	N
Column Aliasing	Y
Convert	N
Table Correlation Names	Y
Different Table Correlation Names	Y
Expressions in Order By	Y
Order By Unrelated	Y
Group by	Y
Group by Unrelated	Y
Group by Beyond Select	Y
Like Escape Clause	Y
Multiple Result Sets	N
Multiple Transactions	Y
Non Nullable Columns	Y
Minimum SQL Grammar	Y
Core SQL Grammar	Y
Extended SQL Grammar	Y
ANSI 92 Entry Level SQL	Y
ANSI 92 Intermediate SQL	N

1.	CURRENCY_CODE	CHAR[3]	1	NO	YES	43 (100.00%)	Value / Length / Count ADF 3 1 ADP 3 1 AED 3 1 AFA 3 1 ALL 3 1 ANG 3 1 AON 3 1 ARA 3 1 ARS 3 1 ATS 3 1	
2.	DESCRIPTION	VARCHAR2 [50]		NO	YES	43 (100.00%)	Value / Length / Count ADF 3 1 ADP 3 1 AED 3 1 AFA 3 1 ALL 3 1 ANG 3 1 AON 3 1 ARA 3 1 ARS 3 1 ATS 3 1	

Data duplication

CURRENCY_CODE and DESCRPTIONS have the same values and it is very likely that one of these columns is redundant.

8.	STATE	VARCHAR2 [50]		YES	NO	53 (17.67%)	Value / Length / Count ON 2 206 TN 2 4 TX 2 4 FM 2 3 ID 2 3 MI 2 3 MN 2 3 NM 2 3 NV 2 3 OR 2 3	
9.	COUNTRY	VARCHAR2 [40]		NO	NO	2 (0.67%)	Value / Length / Count Canada 6 206 United States 13 94	
10.	CURRENT_ADDRESS_IND	CHAR[1]		NO	NO	2 (0.67%)	Value / Length / Count N 1 150 Y 1 150	

Data Dispersion

Verify if data dispersion is as expected. For example, “ON” appears fifty times more than any other STATE. “Canada” and “US” are the only countries in the COUNTRY field though Canada is much more frequent.
CURRENT_ADDRESS_IND has exactly 50% of “Y” and “N” – how likely is it to have that split?

#	11
Table Name	ROLE_PERMISSION
Owner	DBANALYZER_1
Type	TABLE
Row Count	0
Remarks	WARNING: Table does not have Primary Key. WARNING: Table is empty.

#	Column	Type	PK	Null	Unique	Cardinality	Sample Data	Comment
1.	ROLE_ID	NUMBER [22]		NO	N/A	0	N/A	
2.	SYSTEM_ID	NUMBER [22]		NO	N/A	0	N/A	
3.	MODULE_ID	NUMBER [22]		NO	N/A	0	N/A	
4.	WINDOW_ID	NUMBER [22]		NO	N/A	0	N/A	
5.	FUNCTION_ID	NUMBER [22]		NO	N/A	0	N/A	
6.	PERMISSION_TYPE	VARCHAR2 [10]		NO	N/A	0	N/A	

Modeling Issues - Primary Keys

Warnings are shown in RED in case the table requires immediate attention. In this case, the table is missing a Primary Key and it is very likely that the model needs to be verified and corrected.

Also, a table may not have any records. Business logic may need to be verified to confirm whether this table is in fact required.

Column Name	USER_ID
Table Name	LOGIN_HISTORY (1993 records)
Allow NULL	NO
Ordinal Position	2
Data Type	NUMBER[22]
Primary Key Sequence	
Unique	NO
NULL count	0
Cardinality	191 (9.58%)
Distinct values count	191

Sample Data

Value	Length	Count
517157		18
156825		17
601580		17
626540		17
955931		17
113597		16
208770		16
801001		16
251950		15
360806		15

Max String Length	N/A
Max String Sample	N/A
Sample String Data	N/A

Constraint Information

Constraint name	USER HAS LOGIN HISTORY
Referenced column	USER_ID
Update rule	Update of the primary key updates child record(s)
Delete rule	Cannot delete primary key before child record(s) is deleted
Deferrability	Not Deferrable

Remarks	

Detailed column information

This is the optional section in the report. It provides detailed information about database table columns.

"Constraint Information" is particularly useful as it provides information about the referenced column and rules associated to the behavior enforced by the constraint, as well as update, delete rule, and deferability. Many ERD tools will not be able to provide this information.

Note that the table name is an active link which you can use to quickly go back to the table detail section.

#	Column	Type	PK	Null	Unique	Cardinality	Sample Data	Comment
1.	TRANSACTION_ID	NUMBER [22]	1	NO	YES	962 (100.00%)		

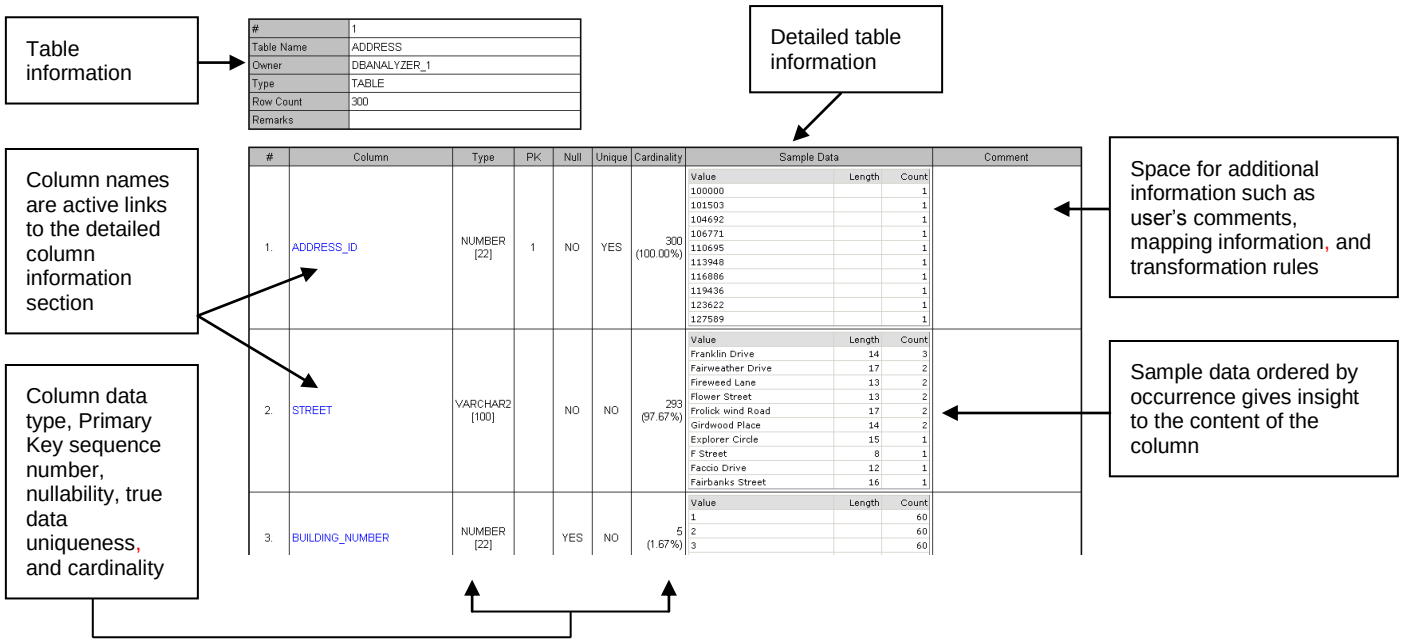
Value	Length	Count
101334		1
101766		1
102639		1
103039		1
103891		1
104054		1
105429		1
106222		1
106749		1
107524		1

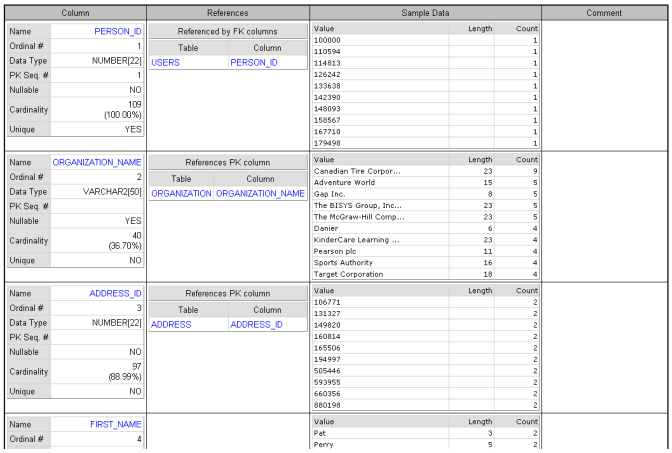
#	Column	Type	PK	Null	Unique	Cardinality	Sample Data	Comment
2.	ITEM_ID	NUMBER [22]	2	NO	YES	962 (100.00%)		

Value	Length	Count
100000		1
100956		1
101731		1
102947		1
104408		1
105031		1
106016		1
107651		1
109902		1
110249		1

Data Anomalies

For this associative table, data sample reveals possible data anomaly - all items are associated only to one transaction.





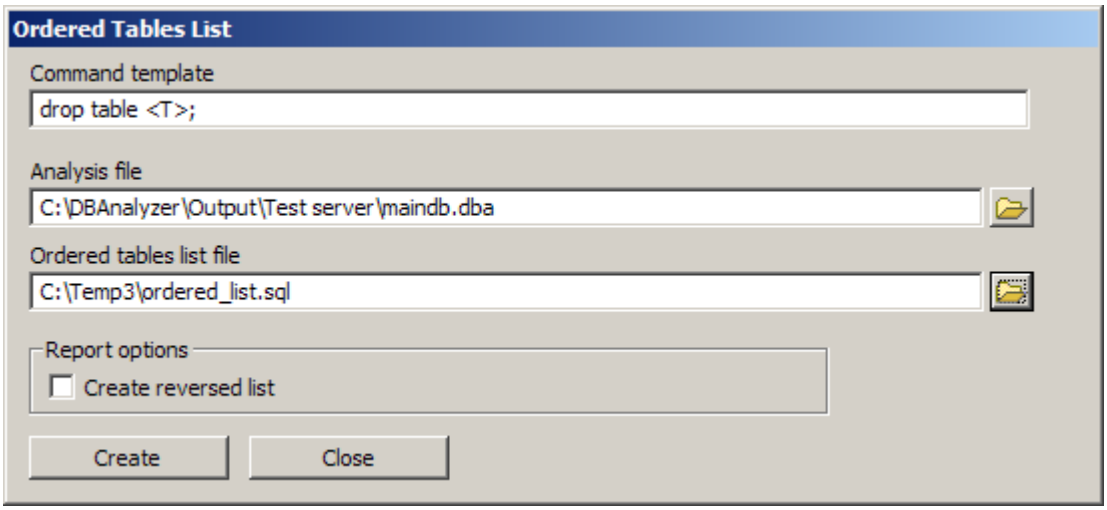
Referential Integrity

This report gives you information about relationships defined between database tables. You can see what columns are referencing the table's Primary Key and what Primary Keys are referenced by the table's Foreign Keys.

Note that all table and column name fields are active links.

Creating Ordered Tables Lists

As mentioned before, this feature is used to produce an ordered table list depending on the relationship between tables. For example, if you want to drop table by table, you would use this method to create a list of tables ordered in the way tables can be dropped, meaning that tables with no children will be dropped first.



This method creates another list with the reverse order from the list created by method "getOrderedTableList". For example, this option can be used to create tables.

File “C:\Temp3\ordered_list.sql” will be created with the following content (see data model on next page):

```
drop table TRANSACTION_ITEM;
drop table USER_PROFILE;
drop table CURRENCY_CODE;
drop table ROLE_PERMISSION;
drop table PERMISSION;
drop table USER_ROLE;
drop table ROLE;
drop table PAYMENT;
drop table PAYMENT_INSTRUMENTS;
drop table LOGIN_HISTORY;
drop table TRANSACTION;
drop table USERS;
drop table PERSON;
drop table ORGANIZATION;
drop table ADDRESS;
drop table ITEM;
```

Note that the database views are ignored when an ordered tables list is created.

Automating Tasks

1. Running Database Analyzer from the command line in Batch Mode

Once you set up your project configuration, it is practical to run Database Analyzer Tasks in Batch Mode. By running DB Analyzer from the command line you can accomplish several tasks without user interaction:

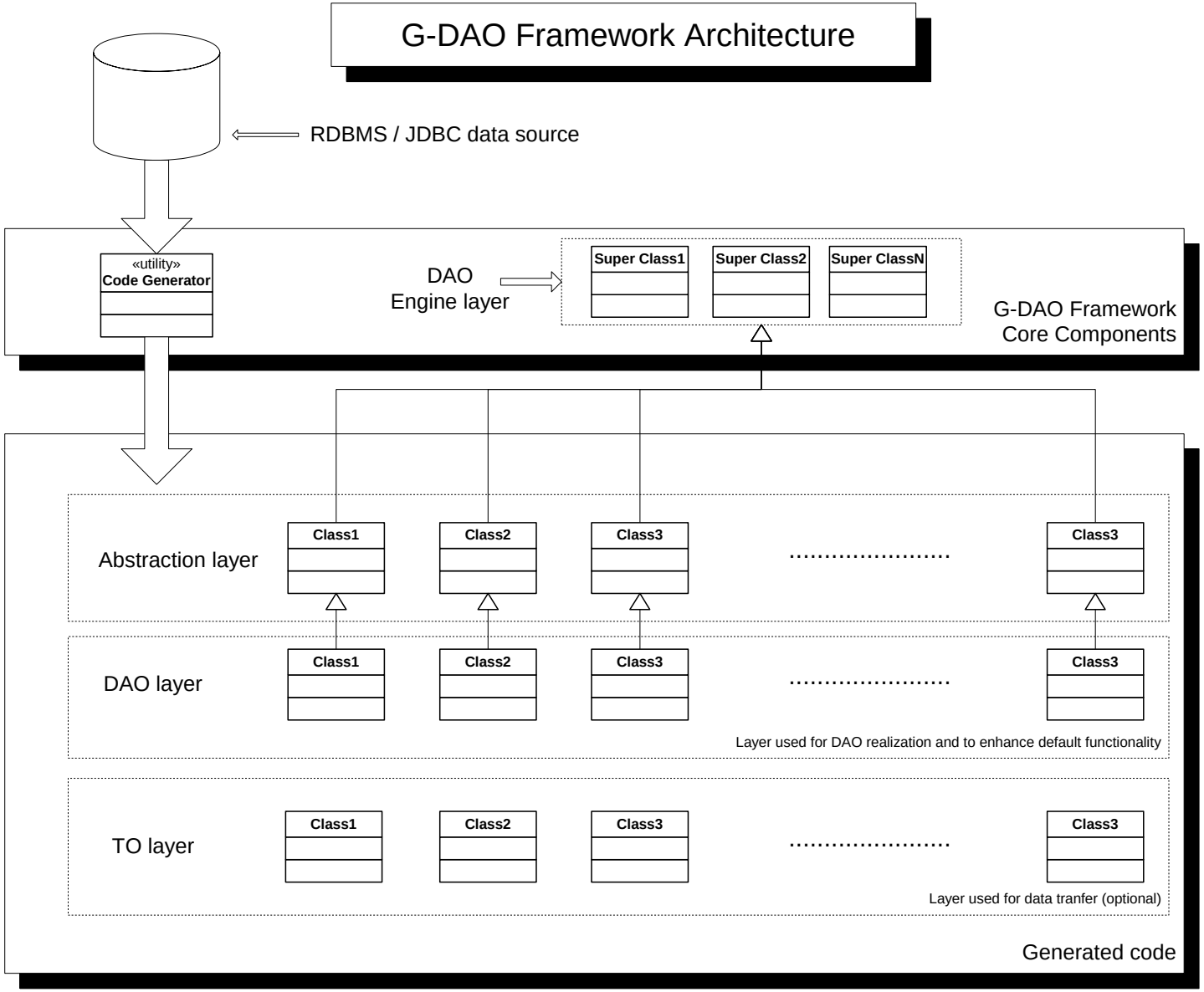
1. Analyze database
2. Generate Java Code
3. Format Generated Java Code
4. Produce Reports

In order to run scripts, you need to install Cygwin.

This is the sample script used to run DB Analyzer in “batch” mode from the command line:

```
\dba\DBA Main\scripts\cygwin_scripts\run.sh
```

GDAO Architecture



Appendix

Now you are ready to create a directory where you are going to put your manually created code. In this example, we will create:

C:\Projects\gdaodemo\myapplication

After analyzing the GDAO DEMO database and generating code in directory C:\Projects\gdaodemo\generated", you can take copies of classes:

c:\Projects\gdaodemo\generated\com\mallocinc\gdaodemo\samples\excel\UserActionLogExcelSample.java

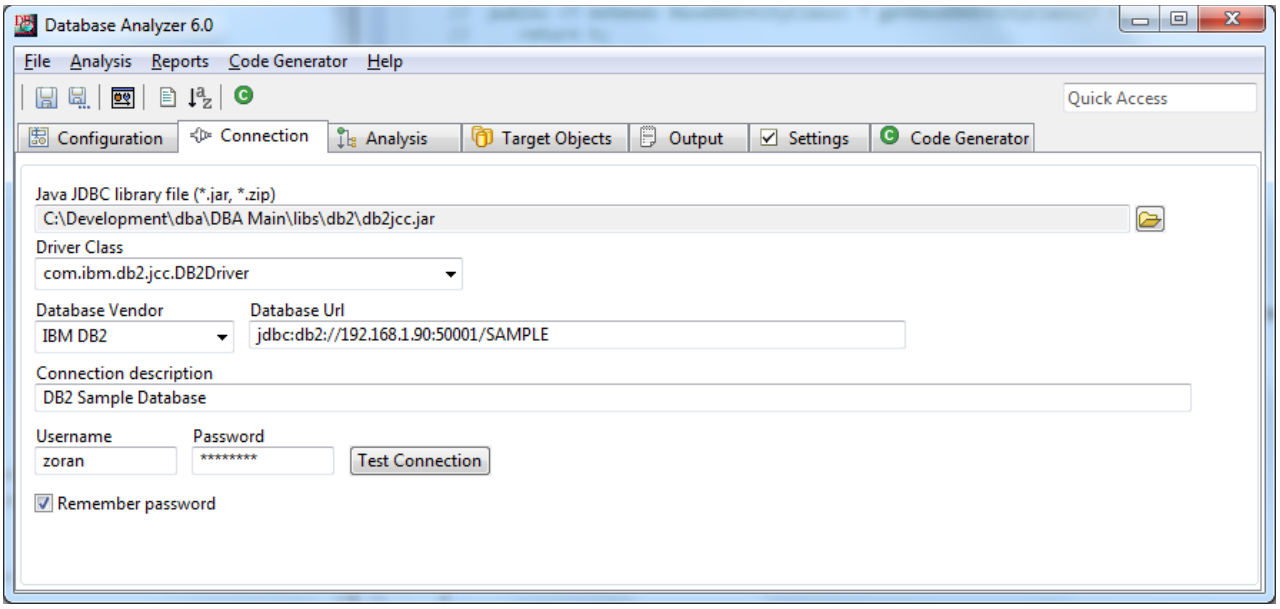
c:\Projects\gdaodemo\generated\com\mallocinc\gdaodemo\samples\excel\UsersExcelSample.java

When you are copying, you should include all directories that form a class package. For example, you should copy "com\mallocinc\gdaodemo\samples\excel\UserActionLogExcelSample.java" to "C:\Projects\gdaodemo\myapplication"

After you are done with copying, you can rename classes according to your project naming convention.

Sample database Connections

2. DB2



Upgrading Eclipse

When you upgrade to the next version of the Eclipse you usuially have to do the following:

1. Double click on “dbanalyzer_64.product” (if you are using 32 bit then “dbanalyzer_32.product”
2. Go to “dependencies” tab
3. Click on “required Plug-ins”
4. Remove invalid (not found) plug ins from the list. The have a small icon indicatior that they are invalid.
5. Save and run the project again